Effective Configuration Management
(9 Things You Should Be Doing)

N.J. Thomas
nthomas@amplify.com

Amplify Education

November 6, 2013
Introduction

- Who I am
- Who you are
- Disclaimer
- Caveats
Proposition 1 - Use Configuration Management

Regardless of how small/large your team is, or how many systems you are managing, pick a modern Configuration Management System and use it.

- Rebuttals:
  - not enough sysadmins
  - not enough machines
  - homegrown system
  - too simple/complex a setup
Modern Configuration Management Systems:
- Ansible
- bcfg2
- CFEngine
- Puppet
- Chef
- Salt

Reasons
- actively developed
- large user/install base
- well documented
- mailing lists/forums/paid support
- pre-existing modules/cookbooks
- multi-platform

LISA ’13 is a great place to make your choice!
Proposition 2 - Use Version Control

Once you have a Configuration Management System chosen, pick a modern version control tool for the backend repository.

- Git, Subversion, Mercurial, etc., are all fine choices.
- NB: presupposes you have a text-based directory tree for backend.
- Reasons:
  - reversibility
  - concurrency
  - annotation
  - multiple branches of development
  - feature/story branching
  - hooks
Proposition 3 - Use A Validation Testing Suite

Before pushing out a newly committed change, make sure it has been run against, and passes, your testing suite.

Tests:

- CM lint
- syntax checking for YAML, XML
- sudoers syntax
- passwd file validity
- nagios -v

Some sites use VCS hooks for this, but a continous integration tool is strongly recommended.
Proposition 4 - Use A Code Review Tool

Before making any commit into your CM repository, make sure it is audited by at least one other peer.

Reasons:
- Github: pull request
- Bitbucket: lightweight code review
- Trac: many plugins (PeerReview, CodeReview)
- Gerrit

Rebuttals:
- small changes (big mistakes)
- too slow (make it lightweight)
- “I know what I’m doing.”
Use Code Review

Code-Review:
- +2 Looks good to me, approved
- +1 Looks good to me, but someone else must approve
- 0 No score
- -1 I would prefer that you didn't submit this
- -2 Do not submit

Cover Message:

Publish Comments  Publish and Submit  Cancel
Use Code Review

<table>
<thead>
<tr>
<th>Change-id</th>
<th>la6507524af8731ea949c4e3c1b500158bf20a360</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>release_als</td>
</tr>
<tr>
<td>Branch</td>
<td>master</td>
</tr>
<tr>
<td>Topic</td>
<td></td>
</tr>
<tr>
<td>Uploaded</td>
<td>2013-10-23 11:04</td>
</tr>
<tr>
<td>Updated</td>
<td>2013-10-23 14:32</td>
</tr>
<tr>
<td>Status</td>
<td>Merged</td>
</tr>
</tbody>
</table>

**Commit Message**

update websync-worker as part of mc22.3.0 for circle-2013 dark release

the traditional accompaniment of webysnc-server is included

<table>
<thead>
<tr>
<th>Reviewer</th>
<th>Verified</th>
<th>Code-Review</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Name or Email or Group  Add Reviewer
Important Note on Testing/Code Review

- Proper use of your testing suite and code review tool will catch 95% of potential problems.

- For the ones that get through, add a test.
Proposition 5 - *Use your CM’s Templating Facilities*

Never manage more than one copy of a file. Use your CM’s templating facilities to manage one file along with a variable properties list.
Use Templating

Don't do this:

restrict 127.0.0.1
restrict default kod nomodify notrap noquery nopeer
server ntp1.qa.example.org
server ntp2.qa.example.org
driftfile /var/lib/ntp/drift

restrict 127.0.0.1
restrict default kod nomodify notrap noquery nopeer
server ntp1.prod.example.org
server ntp2.prod.example.org
driftfile /var/lib/ntp/drift
Use Templating

Do this instead:

{% python
servers = [el.text for el in
    metadata.Properties['ntp.xml'].findall('Server')]
%
restrict 127.0.0.1
restrict default kod nomodify notrap noquery nopeer
{% for server in servers %}
server ${server}
{% end %}
driftfile /var/lib/ntp/drift
Use Templating

The associated properties file:

```xml
<?xml version="1.0" encoding="utf-8"?>
  <Group name="production">
    <Server>ntp1.prod.example.org</Server>
    <Server>ntp2.prod.example.org</Server>
  </Group>
  <Group name="qa">
    <Server>ntp1.qa.example.org</Server>
    <Server>ntp2.qa.example.org</Server>
  </Group>
</Properties>
```
Proposition 6 - Use Your CM for Provisioning

You should be able to use your CM to provision new systems, with little manual intervention.

Notes:

- existing hostclasses/profiles/classes
- ability to scale on demand
- necessary with EC2/cloud services
- common pattern: use pxe/jumpstart/kickstart/bsdinstall/cobbler plus your CM package, then use CM for everything else
Proposition 7 - Use Your CM for Automated Monitoring

For every machine that is provisioned automatically by your CM, you should also be generating monitoring.

Notes:
- use a monitoring system that accepts text-based configs
- always having your new configs go through your testing suite (nagios -v)
- add dummy hosts for sanity checking
Use a CMDB in conjunction with your CM.

"A repository that acts as a data warehouse for information technology (IT) organizations. Its contents is intended to hold a collection of IT assets that are commonly referred to as Configuration Items (CIs), as well as descriptive relationships between such assets." – Wikipedia

Notes:

- some CMs bundle their own CMDB
- use a CMDB that can be programmatically-queried
- can use the CMDB to pull into CM and vice versa
- if using homegrown, make it lightweight
- don’t use a spreadsheet
- don’t use the CM as a CMDB
<table>
<thead>
<tr>
<th>Proposition 9 - Use Your CM for DevOps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Your CM for DevOps tasks. This includes, but is not limited to, software/product deployments, security auditing, quality testing, etc.</td>
</tr>
</tbody>
</table>

"DevOps (a portmanteau of development and operations) is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) professionals. DevOps is a response to the interdependence of software development and IT operations. It aims to help an organization rapidly produce software products and services.” – Wikipedia
Questions?